TUNING PARAMETERS OF DYNAMIC MATRIX CONTROL (DMC)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology in

Electronics and Instrumentation Engineering

By

NIKHIL VATSA

ROLL NUMBER: 107EI031

Department of Electronics & Communication Engineering

National Institute of Technology

Rourkela

2011

TUNING PARAMETERS OF DYNAMIC MATRIX CONTROL (DMC)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology in

Electronics and Instrumentation Engineering

Under the Guidance of

Prof. T K DAN

By

NIKHIL VATSA

ROLL NUMBER: 107EI031

Department of Electronics & Communication Engineering

National Institute of Technology

Rourkela

2011

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled "**Tuning Parameters of Dynamic Matrix Control**", submitted by Mr. NIKHIL VATSA in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in 'ELECTRONICS & INSTRUMENTATION' Engineering at the National Institute of Technology (NIT), Rourkela is an authentic work carried out by him under my supervision.

Date:

Prof. Tarun Kumar Dan

Department of Electronics and Communication Engg.

National Institute of Technology, Rourkela

Rourkela-769008

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude and sincere thanks to my respected supervisor Prof. T.K DAN for the guidance, insight, and support that he has provided throughout the course of this work. The present work would never be possible without his vital inputs and mentoring,

I would like to thank all my friends, faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T Rourkela for their extreme help throughout my course of study at this institute.

Nikhil Vatsa (107EI031)

CONTENTS

ABSTRACT

Model predictive control (MPC) was developed to meet control challenges of Chemical Industries. With the passage of time, it has become the one of the most effective advanced control technique for a wide range of industries. The advantages of MPC are most evident when it is used as a multivariable controller integrated with an optimizer. Dynamic Matrix Control (DMC) was the first Model Predictive Control (MPC) algorithm developed by Shell Oil Company in 1970s. The advantages of these methods have already been proven and these methods have been found to work satisfactorily for long durations of time. DMC is available in all industrial process control systems and on a number of control simulation platforms. DMC is particularly advantageous for Multiple Input Multiple Output (MIMO) Systems.

This work deals with DMC for an unconstrained Single Input Single Output (SISO) system. Finite step Response (FSR) has been used for model prediction. Quadratic objective function, based upon the square of differences between predicted outputs and set-point, has been used. DMC is subsequently applied for the optimization of the objective function. It has shown that DMC has got inbuilt compensation for systems with dead time and systems with non-minimum phase. The effect of Tuning Parameters of Dynamic Matrix Control on the response of the system has been taken up. Various models have been simulated on **MATLAB** for this purpose. Effect of Noise on the system has been also studied. Modeling of a DC motor has been done at the end and an optimal set of Tuning Parameters has been found out. Further work is required to be done towards the application of this method to constrained systems and Multiple Input Multiple Output (MIMO) systems.

List of Figures

List of Tables

Chapter 1

INTRODUCTION TO MODEL PREDICTIVE CONTROL (MPC)

1.1 Model Predictive Control (MPC) Background

The term “MPC” does not itself designate a specific control strategy but a very wide range of control methods which make an explicit use of a model of the process to obtain the control signal by optimization of an objective function. These design methods lead to linear controllers which have practically same structure and present adequate degrees of freedom. The objective function, also known as the cost function, is based on predicted outputs over a prediction horizon of P time steps. The minimization of the cost function is done by a selection of manipulated process variables over a control horizon of M control moves. The selection of M control moves does not mean that all of them are implemented. As a matter of fact, only the first control move is implemented. After it, model correction for predicted output at the next time step is carried out. The whole optimization process is done again at the new step and the process is carried on. The basic structure of MPC is shown below:

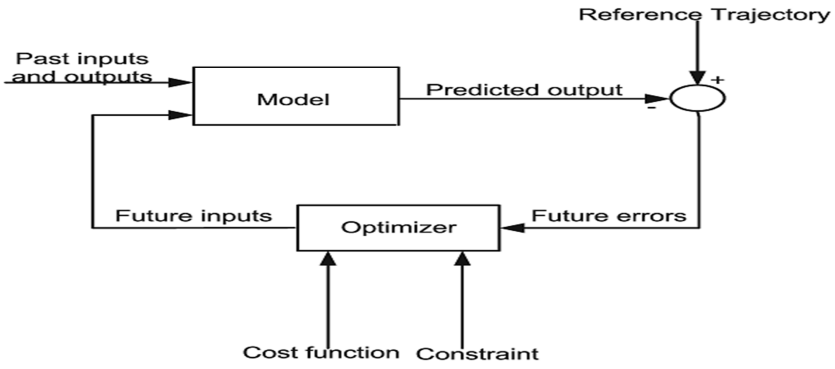


Figure1.1: Basic structure of MPC [1]

1.2 Basic Components of MPC:

A simple MPC controller three basic components:

- A process model that predicts the process output in the future up to the prediction horizon P.

- A future trajectory of the set point.
- A control algorithm for computing a control action based on the error vector as the difference between the future trajectories of the set point and the predicted process output. At each instant the prediction horizon is displaced towards the future which involves the application of the first controls signal of the sequence at each step.

The **receding horizon** concept can be better understood through the following figure

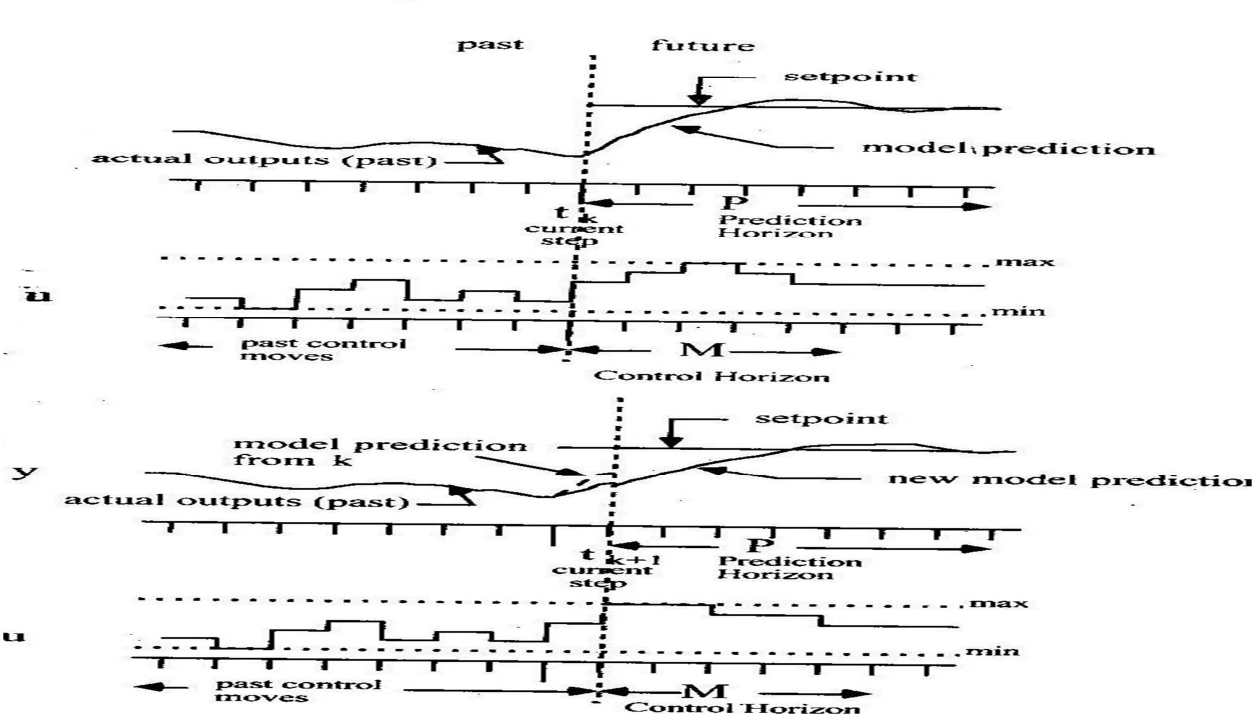


Figure 1.2: Receding Horizon Strategy [2]

1.3 Advantages of MPC:

- It is an easy to tune method and is particularly to staff with only a limited knowledge of control.

- It can be used to control a great variety of processes from those with a relatively simple dynamics to other more complex ones, including systems with long delay times or of non-minimum phase or unstable ones.
- It handles multivariable control problems easily.
- It can handle non-minimal phase and unstable process.
- It intrinsically has compensation for dead time.
- It allows operation closer to constraints, hence increases profit.
- It is very useful when future references are known (robotics or batch processes).
- It has plenty of time for on-line computations.
- It can take account of actuator limitations.
- It handles structural changes and is a totally open methodology based on certain principles

1.4 Limitations of MPC:

- Despite the fact that the resulting control law used in MPC is much easier, its derivation has got far much greater complexity than a traditional PID controller.
- In case of adaptive control, the computations regarding the control law has to be done at every sampling interval. Today's process computers despite, having no problems in carrying out these computations, have to be used often for purposes (like monitoring etc.) other than computations.

1.5 MPC versus Feedback Control:

MPC	Feedback Control
A predicted error vector is applied to the MPC controller algorithm	The scalar values of recent errors are used in a feedback controller
The error vector for an MPC controller is computed as the corrected model prediction subtracted from the future set-point values	The error in a feedback controller is the measurement subtracted from the set-point value

Table 1.1: MPC versus Feedback Control

1.6 Types of MPC:

Linear MPC	Nonlinear MPC
1. Uses linear model: $\dot{x} = Ax + Bu$	1. Uses nonlinear model: $\dot{x} = f(x,u)$
2. Quadratic cost function: $F = x^T Q x + u^T R u$	2. Cost function can be nonquadratic: $F(x,u)$
3. Linear constraints: $Hx + Gu < 0$	3. Nonlinear constraints: $h(x,u) < 0$
4. Quadratic program	4. Nonlinear program

Table 1.2: Types of MPC [3]

Chapter 2

OPTIMIZATION AND PROCESS MODELS

2.1 Objective (Cost) Function:

Cost function or objective function is a measure of performance of the process model. The term optimization implies a best value for some type of performance criterion. This performance criterion is known as the objective function. There are several choices for the objective function. Two of the most commonly used objective functions are given below

2.1.1 Quadratic Objective Function: It is a "sum of squares" of the difference between the set-points & the model predicted outputs and the control moves. A control move is the change in control action from one step to another step. For a P-step prediction horizon and a M-step control horizon, the quadratic objective function is given below

$$\Phi = \sum_{i=1}^{p} \left(r_{k+i} - y_{k+i}\right)^2 + + w\left(\sum_{i=0}^{i=M-1} (\Delta u_{k+i})^2\right)$$

where:

w= weight of the control move

Δu =change in the manipulated variable from one sample time to another sample time

r is the set-point and y is the model predicted output

This objective function penalizes larger error more than smaller errors and hence seems more logical.

2.1.2 Absolute value Objective Function: It simply takes the sum of absolute values of predicted errors and control moves. It is given as:

$$\Phi = \sum_{i=1}^{p} I\left(r_{k+i} - y_{k+i}\right) I + + w\left(\sum_{i=0}^{i=M-1} I \Delta u_{k+i} I\right)$$

The limited use of absolute value objective function is due to the fact that it results into linear programming problem.

2.2 MPC Controller Generic Configuration:

MPC based controller, like all the controllers, controls some process model which is shown in the following diagram.

In MPC, the process inputs are Manipulated Variables (MVs) and measured Disturbance Variables (DVs). The process outputs are CVs and auxiliary or constraint variables (AVs) [4].

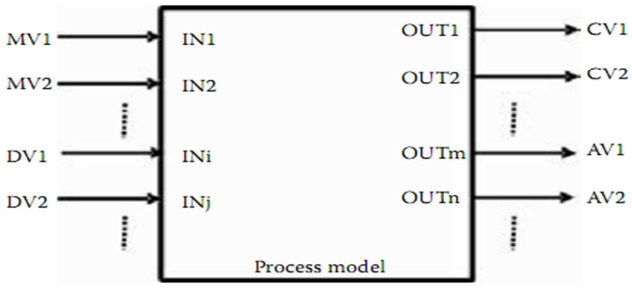


Figure 2.1: Multivariable MPC-controlled generic process configuration [5]

- **Manipulated variables** are outputs of the MPC controllers. MPC Controller makes several future moves on the manipulated variable for the sake of optimization. Only the first control move is implemented and this procedure is repeated every time step on the prediction horizon. The control horizon is the number of future manipulated variable moves that are taken into account in developing the optimal MPC solution.
- **Measured disturbances** are also inputs to the process. But, they are not managed by MPC.
- **Controlled variables** are process outputs kept at specific set points (targets) or within specified ranges.
- **Constraint variables** are a type of controlled variables with only range control and no set points.

2.3 Process Models:

Process models predict the future values of controlled variables for a number of discrete time steps ahead. The process model is the basis for MPC technology. There are different types of process models are used in calculating the predicted values of the process outputs. For linear MPC applications, most algorithms use one of the three models namely state space model, step response model and impulse response model. It is a good practice to make use of discrete models for the output prediction whenever possible because MPC is a discrete time domain control algorithm. The step and impulse response models find frequent use in MPC algorithms.

2.3.1 State Space Model: A state space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations. The state space representation (also known as the "time-domain approach") provides a convenient and compact way to model and analyze systems with multiple inputs and outputs. "State space" refers to the space whose axes are the state variables. The state of the system can be represented as a vector within that space. The most general state-space representation of a linear system with p inputs, q outputs and n state variables is written in the following form [6]:

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$
$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

where:

$\mathbf{x}(\cdot)$ is called the "state vector", $\mathbf{x}(t) \in \mathbb{R}^n$;
$\mathbf{y}(\cdot)$ is called the "output vector", $\mathbf{y}(t) \in \mathbb{R}^q$;
$\mathbf{u}(\cdot)$ is called the "input (or control) vector", $\mathbf{u}(t) \in \mathbb{R}^p$;
$A(\cdot)$ is the "state matrix", $\dim[A(\cdot)] = n \times n$,
$B(\cdot)$ is the "input matrix", $\dim[B(\cdot)] = n \times p$,
$C(\cdot)$ is the "output matrix", $\dim[C(\cdot)] = q \times n$,
$D(\cdot)$ is the "feedthrough (or feedforward) matrix" (in cases where the system model does not have a direct feedthrough, $D(\cdot)$ is the zero matrix), $\dim[D(\cdot)] = q \times p$,
$\dot{\mathbf{x}}(t) := \dfrac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t)$

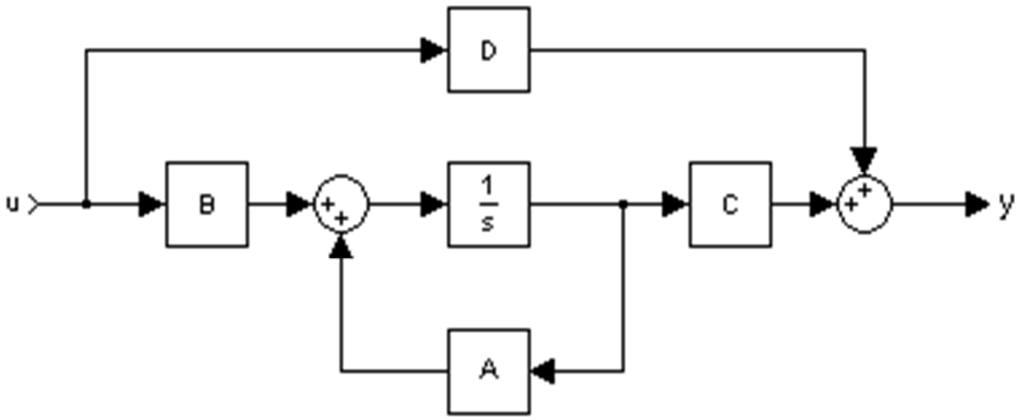


Figure 2.2: Block diagram of State Space model [7]

2.3.2 Finite Step Response (FSR) model: FSR models are obtained by making a unit step input change to a process operating at steady state. The model coefficients are simply the output values at each time step. Here, s_i represents the step response coefficients for the i^{th} sample time after the unit step input change. If a non-unit step change is made, the output is scaled accordingly. The step response model is the vector of step response coefficients

$$S=[\ s_1\ s_2\ s_3\ s_4\ s_5\ \ldots\ s_N]^T$$

The model length N should be long enough so that the coefficients values are relatively constant (they saturate).

2.3.3 Finite Impulse Response (FIR) model: A unit pulse is applied to the manipulated input, and the model coefficients 'h_i' are simply the values of the outputs, i.e, the i^{th} impulse response coefficients. There is a direct relationship between step and impulse response models:

$$h_i = s_i - s_{i-1}$$

$$Si = \sum_{j=1}^{i} h_j \ .$$

It should be noted that there are two major limitations to step and impulse response models. They can only be used to represent open-loop stable processes, and they require a large number of parameters (model coefficients) compared to state space and transfer function models.

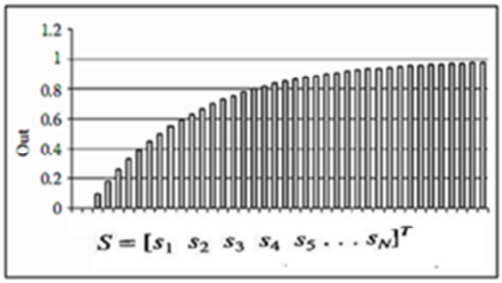


Figure 2.3: FSR Model

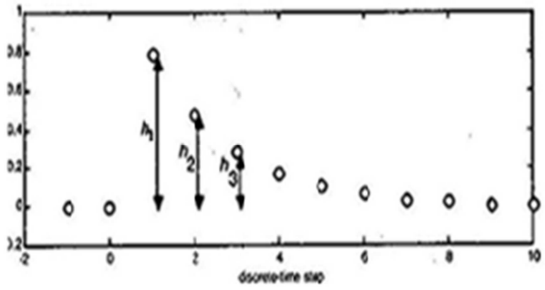


Figure 2.4: FIR Model

Chapter 3

DYNAMIC MATRIX CONTROL

3.1 Dynamic Matrix Control (DMC) Introduction:

- MPC includes a series of algorithms among which the Dynamic Matrix Controller (DMC) is one of the most important ones.
- DMC were developed for Cutler and Ramaker [8], and it has been used in the industrial world, mainly in the petrochemical industries.
- DMC is a linear control technique where the process is represented by a first order plus dead time (FOPDT) model.
- The model response to a unit step change is used to predict the future response of the dependent variables and formulates a series of control actions for all the independent variables. The actions are selected to minimize the error of the process on the time horizon.

3.2 Mathematical Formulation:

For a Prediction horizon of length P and control horizon of length M, the objective function can be found out through following method**.** Based on the step response model we have

$$y_k = \sum_{i=1}^{\infty} s_i \Delta u_{k-i} \qquad \text{----(1)}$$

However, since the step response coefficients saturate after N steps we have,

$$S_N = S_{N+1} = S_{N+2} = \ldots\ldots\ldots\ldots\ldots\ldots \qquad \text{----(2)}$$

Putting (2) into (1), we have:

$$y_k = \sum_{i=1}^{N-1} s_i \Delta u_{k-i} + s_N \sum_{i=N}^{i=\infty} \Delta u_{k-i}$$

$$\Rightarrow y_k = \sum_{i=1}^{N-1} s_i \Delta u_{k-i} + s_N (u_{k-N}) + d_k \qquad \text{----(3)}$$

Now, since the model predicted output is unlikely to be equal to the actual measured output, we introduce an additive disturbance (d_k) for corrected model prediction.

The corrected prediction is $y^c_k = y_k + d_k$ ----(4)

Putting (4) into (3),we have

$$y^{c}_{k+1} = \sum_{i=1}^{N-1} s_i \Delta u_{k-i+1} + s_N (u_{k-N+1}) + d_{k+1}$$

$$\Rightarrow y^{c}_{k+1} = s_1 \Delta u_k + \sum_{i=2}^{N-1} s_i \Delta u_{k-i+1} + s_N (u_{k-N+1}) + d_{k+1}$$

$$\Rightarrow y^{c}_{k} = \sum_{i=1}^{i=j} s_i \Delta u_{k-i+j} + \sum_{i=j+1}^{N-1} s_i \Delta u_{k-i+j} + s_N (u_{k-N+j}) + d_{k+j}$$

$$\Rightarrow \quad y^{c}_{k+j} = \sum_{i=1}^{i-j} s_i \Delta u_{k-i+j} + \sum_{i-j+1}^{N-1} s_i \Delta u_{k-i+j} + s_N (u_{k-N+j}) + d_{k+j}$$

Effect of current and future moves

effect of past moves

correction term

---(5)

We take the following two assumptions:

1) Correction term is constant in future("**constant additive disturbance assumption**")
   $d_{k+j} = d_{k+j+1} = \ldots\ldots\ldots$

2) Since there are no control moves beyond M-steps,
   $\Delta u_{k+M} = \Delta u_{k+M+1} = \ldots\ldots\ldots \Delta u_{k+P-1} = 0$

Now for a prediction horizon of P steps and a control horizon of M steps, the matrix-vector form representation is:

$$\begin{pmatrix} yk+1 \\ yk+2 \\ . \\ . \\ yk+j \\ . \\ . \\ yk+P \end{pmatrix} = \begin{pmatrix} s1 & 0 & & 0.. & & ..0 \\ & & . & & & \\ & & . & & & \\ sj & sj-1 & sj-2 & 0 & \dots & sj-M+1 \\ . & & & & & \\ . & & & & & \\ sP & sP-1 & sP-2 & 0 & \dots & sP-M+1 \end{pmatrix} \begin{pmatrix} \Delta uk \\ \Delta uk+1 \\ . \\ . \\ . \\ . \\ . \\ . \\ \Delta uk+M-1 \end{pmatrix} +$$

Px1:Predicted output	PxM :Dynamic Matrix	Mx1:current & future control moves

$$\begin{pmatrix} s2 & s3 & .. & & ..sN-1 \\ & & . & & \\ & & . & & \\ sj+1 & sj+2 & \dots\dots\dots. & sN-1 & \dots\dots\ 0 \\ . & & & & \\ . & & & & \\ sP+1 & sP+2 & \dots\dots\dots. & 0 & \dots\dots\ 0 \end{pmatrix} \begin{pmatrix} \Delta uk-1 \\ \Delta uk-2 \\ . \\ . \\ \Delta uk-j \\ . \\ . \\ \Delta uk-N+2 \end{pmatrix} +$$

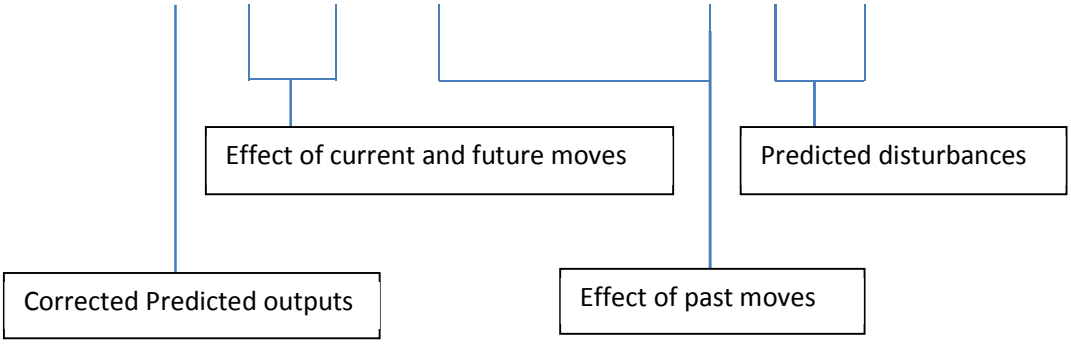
P*(N-2): S_{past} Matrix	(N-2)x1: Past Control moves

$$S_N \begin{pmatrix} uk-N+1 \\ uk-N+2 \\ . \\ . \\ uk-N+j \\ . \\ . \\ uk-N+P \end{pmatrix} + \begin{pmatrix} dk+1 \\ dk+2 \\ . \\ . \\ dk+j \\ . \\ . \\ dk+P \end{pmatrix}$$

Px1: Past inputs u_p	Px1: predicted disturbance

The above expression can be written using matrix vector notation as [9]:

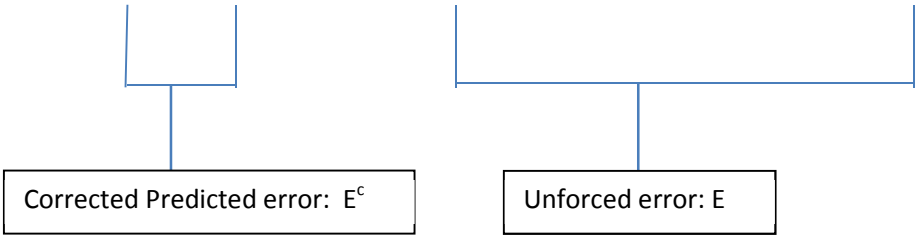
$$Y^c = S_f\Delta u_f + S_{past}\Delta u_{past} + s_N u_P + \hat{d} \quad \text{----(6)}$$



In the equation (6), the corrected prediction is composed of a "**free response**" (output predictions in absence of any control moves) and a "**forced response**"(contributions of present and future control moves).

The difference between the reference trajectory and corrected output prediction will be nothing but corrected predicted error

$$r - Y^c = -S_f\Delta u_f + r - [S_{past}\Delta u_{past} + s_N u_P + \hat{d}]$$



$$E^c = E - S_f\Delta u_f \quad \text{----(7)}$$

We already know that the quadratic or least-square cost function can be written as:

$$\Phi = \sum_{i=1}^{p} (e^c_{k+i})^2 + w\left(\sum_{i=0}^{i=M-1} (\Delta u_{k+i})^2\right) \quad \text{----(8)}$$

Here,

$$\sum_{i=1}^{P} (e^{c}_{k+i})^2 = \begin{bmatrix} e^{c}_{k+1} e^{c}_{k+2} .. e^{c}_{k+P} \end{bmatrix} \begin{bmatrix} e^{c}_{k+1} \\ e^{c}_{k+2} \\ . \\ . \\ e^{c}_{k+P} \end{bmatrix} = (E^{c})^{T} E^{c} \qquad \text{----(9)}$$

$$w\sum_{i=0}^{M-1} (\Delta u_{k+i})^2 = w \begin{bmatrix} \Delta u_{k+1} \Delta u_{k+2} .. \Delta u_{k+M-1} \end{bmatrix} \begin{bmatrix} \Delta u_{k+1} \\ \Delta u_{k+2} \\ . \\ . \\ \Delta u_{k+M-1} \end{bmatrix}$$

$$= \begin{bmatrix} \Delta u_{k+1} \Delta u_{k+2} .. \Delta u_{k+M-1} \end{bmatrix} \begin{bmatrix} w & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & w \end{bmatrix}_{MxM} \begin{bmatrix} \Delta u_{k+1} \\ \Delta u_{k+2} \\ . \\ . \\ \Delta u_{k+M-1} \end{bmatrix}$$

$= \Delta u_f^{T} W \Delta u_f$ ----(10)

Putting (9) & (10) into (8) and using (7) we can write the objective function as:

$$\Phi = (E\text{-}S_f + \triangle u_f)^T (E\text{-}S_f \triangle u_f) + (\triangle u_f)^T W \triangle u_f \qquad \text{----(11)}$$

$E|_{Px1}$: Unforced Error

$S_f|_{PxM}$: Dynamic Matrix

$\Delta u_f|_{Mx1}$: Current and future control moves

$W|_{MxM}$: Diagonal matrix containing weight 'w' at its diagonal

The optimization of Φ for an unconstrained system yields [10]

$\Delta uf = (S_f^T S_f + W)^{-1} S_f^T E$ → Unforced Error

K:Controller gain Matrix

----(12)

Because only current control move is implemented, we use the first row of K Therefore, we get

$$\Delta u_k = K_1 E \qquad \text{----(13)}$$

K_1 represents the first row of K matrix

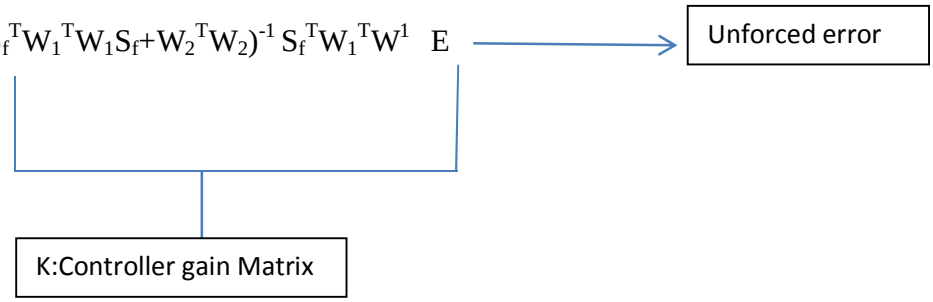
3.2.1 Assigning weights to both Error vector & Control Moves

Assigning weights to both error vector and control moves, we can rewrite the equation (8) as:

$$\Phi = w1 \sum_{i=1}^{p} (e^c_{k+i})^2 + w2 \left(\sum_{i=0}^{i=M-1} (\Delta u_{k+i})^2 \right) \qquad \text{----(14)}$$

Proceeding in the similar fashion as the earlier case, the optimization of the cost function yields us the following control law**:**

$$\Delta uf = (S_f^T W_1^T W_1 S_f + W_2^T W_2)^{-1} S_f^T W_1^T W^1 \quad E$$



----(15)

$E|_{Px1}$: Unforced Error

$S_f|_{PxM}$: Dynamic Matrix

$\Delta u_f|_{Mx1}$: Current and future control moves

$W_2|_{MxM}$: Diagonal matrix containing weight 'w2' at its diagonal

$W_1|_{PxP}$: Diagonal matrix containing weight 'w1' at its diagonal

3.3 DMC Tuning Parameters:

3.3.1 Introduction to Parameters [According to the control law assigned in Equation (15)]:

The Design of any controller involves certain design parameters which are tuned to certain values to get the desired response. Following are the tuning parameters in DMC [11]:

- **Model Length N**: N is an important factor. It affects the step response coefficients as well as the disturbance response coefficients. It is related to the sampling period T by the relation that $T=N\Delta t$ where Δt is the sampling interval. The lowest value of this sampling period is generally limited by computer capacity and computational speed, and N is generally taken between20-70.

- **Prediction Horizon length P**: P determines how far into the future the control objective reaches and thus includes the main dynamic characteristics of the target. Increasing P makes the control more accurate but increases the computation.

- **Control horizon length M**: M determines the number of the control actions calculated into the future. Small value of M makes the controller insensitive of noise. The less M is useful

for controlling the stability of the system while larger M results in excessive control action and increases the flexibility, but it may lead to instability.

- **Error weight matrix W_1**: W_1 is diag(w1,w1 …P times……. w1). The selection of w1 determines the corresponding error term in the optimized control law.

- **Control weight matrix W_2**: W_2 is diag(w1,w1 …M times……. w1). Here,w1 is the weight assigned to the control moves.

3.3.2 Parameter Tuning Step [12]:

- The model horizon N should be selected so that $N\Delta t \geq$ open loop settling time. Δt is the time interval between successive intervals. Normally, the value of N is taken from 20-70. Now, if τ is the dominant Time Constant of the system, then the settling time is around 5 τ. If we take around 50 step response coefficients, then $5\ \tau \approx 50\Delta t$ which means the sampling interval Δt is nearly one-tenth of τ. he model coefficient changes should be as smooth as possible.

- P should cover the main dynamic parts of the step response and increasing it results in more conservative control action but increases computational effect.

- For monotonous dynamic characteristics M =1−2 ; for oscillation dynamic characteristics M=4-8.

- Generally W_1=I (Identity Matrix) and W_2 =pI (Identity Matrix) with p being a constant. Larger values of p penalize the control moves making the system response sluggish. If P>>M, then weight $p \approx 0$. If not, then control moves tend to be aggressive and hence suitable weight p is assigned.

Chapter 4

SIMULATIONS IN MATLAB

4.1 System with Dead Time:

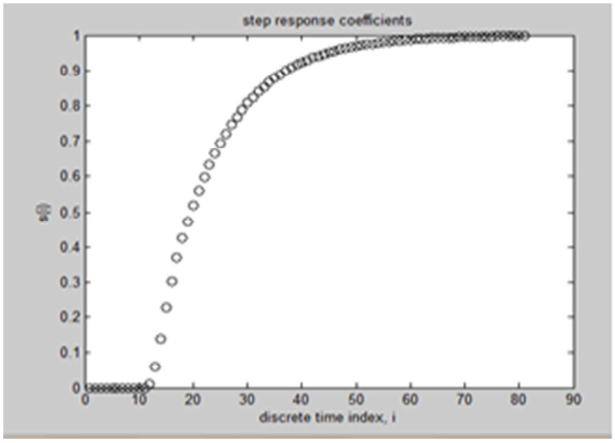
Time delays are found in processes due to a variety of reasons ranging from transport delays to measurement sample delays. Such a system is said to have a dead time. In Laplace Domain, a system with delay time τ is represented by multiplying the Laplace transform of the non-delayed system with $e^{-(\tau s)}$. Let's take the following system having certain delay time:

Plant = $e^{-5s}/(12s^3+13s^2+12s+1)$

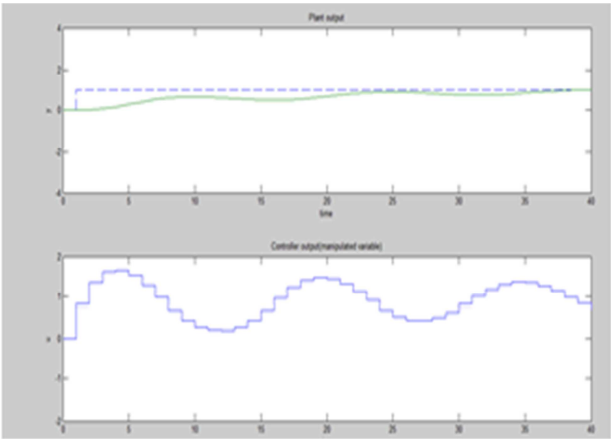
By using the stepinfo command in Matlab, we get the following information about step-response of the system:

Rise Time: 23.651
Settling Time: 48.7223
Settling Min: 0.9005
Settling Max: 1.0000
Overshoot: 0
Undershoot: 0
Peak: 1.0000
Peak Time: 126.0046

Now, after applying DMC to the above plant, the step response and DMC controlled plant output is shown in the follwing simulation



Simulation 4.1: Step Response Coefficients



DMC controlled plant output (green) and Control moves (blue)

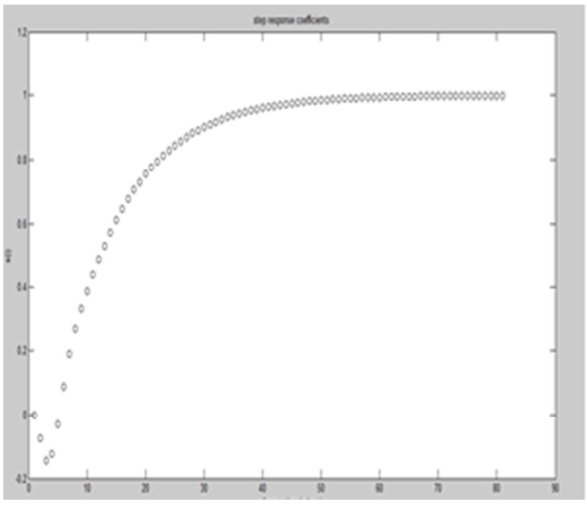
As we can see from the above simulation, the effect of dead-time in the plant response has been successfully negated.

4.2 System with Non-Minimum Phase:

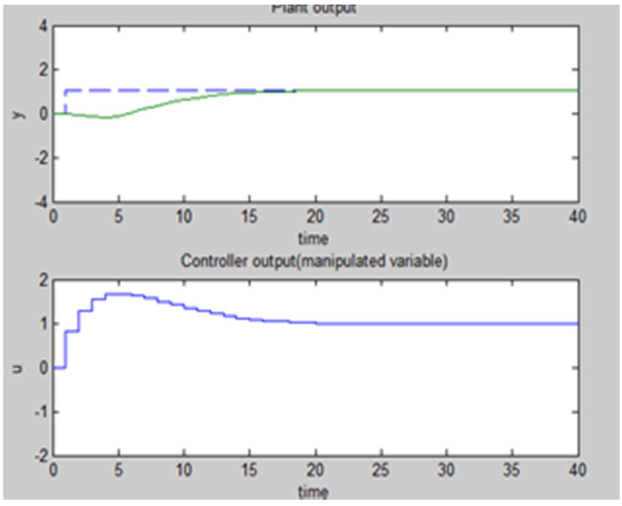
Systems having Transfer function having zeros in right hand plane exhibit Non-minimum phase, i.e. the initial response of the system is towards negative direction even though the final response attains a positive steady state value. Let's take the following system showing this characteristic:

Plant = $(-3s+1) / (12s^3+13s^2+12s+1)$

By using the stepinfo command in Matlab, we get the following information about step-response of the system:
Rise Time: 23.6723
Settling Time: 44.8747
Settling Min: 0.9008
Settling Max: 1.0000
Overshoot: 0
Undershoot: 14.7366
Peak: 1.0000
Peak Time: 114.4445
Now, after applying DMC to the above plant, the step response and DMC controlled plant output is shown in the following simulation



Simulation 4.2: Step Response Coefficients



DMC controlled plant output (green) and Control moves (blue)

As we can see from the above simulation, the non-minimum phase behavior has been removed by designing a DMC based Controller.

4.3 Water-Heater:

4.3.1 Introduction:

Here, we have to heat cold water by the means of a gas burner. The outlet temperature which is nothing, but the temperature of the heated water, depends on the energy added to the water through the gas burner [13]. The outlet temperature can be controlled by manipulating the flow of gas through a control valve which provides a variable restriction to the flow of gas. The position of control Valve is varied for this manipulation.

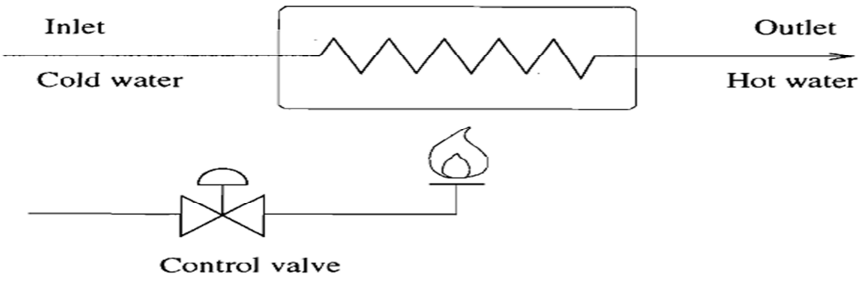


Fig 4.1: Water Heater [14]

The process transfer function [15] (in discrete time domain) relating the output temperature and control valve position is given as:

$$G(z) = \frac{0.2713z^{-3}}{1 - 0.8351z^{-1}}$$

Matlab code :

%N=50,P=10,M=1,w1=1,w2=0

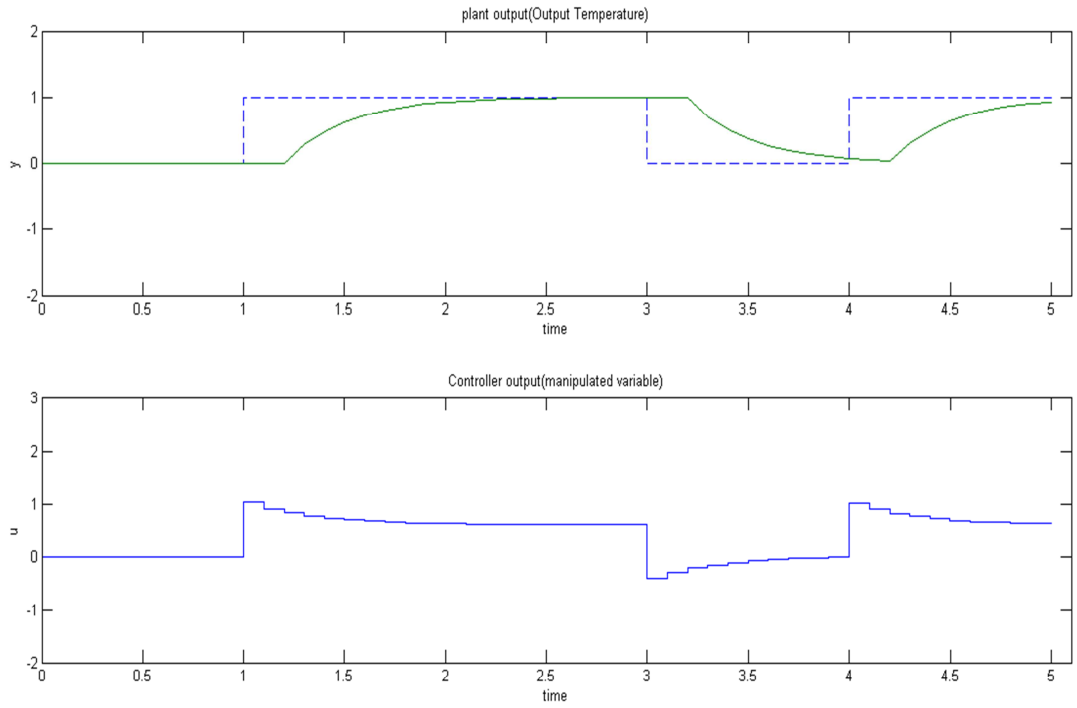
```
clc
clear all
P=10;%prediction horizon
M=1;%control horizon
N=50;%model length
w2=0;%control weight
w1= 1;%error weight
ysp=1;%output set point from 0
timesp=1;%time of set point chang
delt=0.1;%sampling time interval
tfinal=5;%final simulation time
%define time
```

```
tvec=0:delt:tfinal;
ksp=fix(timesp/delt);
kfinal=length(tvec);
%define set point vector
r=[zeros(1,ksp),ones(1,(kfinal-ksp)/2)*ysp,zeros(1,(kfinal-ksp)/4)*ysp,ones(1,(kfinal-
ksp)/4+1)*ysp];
z= tf('z',delt);  plant = (.2713)/(z^3-0.8351*z^2)
%/////////////////define model here//////////////////////////////////
%assumption plant = model
model=plant;
% [numm,denm,tm]=tfdata(plant);
numm = get(model,'num'); numm = numm{:}; % Get numerator polynomial
denm = get(model,'den'); denm = denm{:}; % Get denominator polynomial
%define step response coefficient matrix
s=step(model,0:delt:N*delt);
%Calculation of Sp
for i=1:P
  for j=1:N-2
    if(i+j<=N-1)
     Sp(i,j)=s(i+j+1);
          else
        Sp(1,j)=0;
     end
   end
end
 %Calculation of Dynamic Matrix Sf
 for i=1:P
   for j=1:M
      if i+1-j>0
      Sf(i,j)=s(i+2-j);
      else
          Sf(i,j)=0;
      end
   end
    end
Sf
%obtain W1 and W2 matrix
 W1=w1*eye(P,P);
 W2=w2*eye(M,M);
%obtain Kmat where Kmat=inv(Sf'*W1'*W1*Sf + W2'*W2)*Sf'*W1'*W1;
Kmat=inv(Sf'*W1'*W1*Sf + W2'*W2)*Sf'*W1'*W1;
%piant initial conditions
ndenm=length(denm)-1;
nnumm=length(numm)-1;
umpast=zeros(1,nnumm);
ympast=zeros(1,ndenm);
```

```
uinit=0;
yinit=0;
%initialize input vector
u=ones(1,min(P,kfinal))*uinit;
dist(1)=0;
y(1)=yinit;
dup=zeros(1,N-2);
for k=1:kfinal
    [m,p]=size(Kmat);
   for i=1:p
      if k-N+i>0
         uold(i)=u(k-N+i);
      else
         uold(i)=0;
               end
   end
   dvec=dist(k)*ones(1,p);
   rvec=r(k)*ones(p,1);
   y_free=Sp*dup' + s(N)*uold'+dvec';
   E=rvec-y_free;
   delup(k)=Kmat(1,:)*E;
   if k>1
            u(k)=u(k-1)+delup(k);
   else
      u(k)=delup(k)+uinit;
   end
   %plant equations
   umpast=[u(k),umpast(1,1:length(umpast)-1)];
   y(k+1)=-denm(2:ndenm+1)*ympast'+numm(2:nnumm+1)*umpast';
   ympast=[y(k+1),ympast(1:length(ympast)-1)];
   %model prediction
   if k-N+1>0
      ymod(k+1)=Sf(1,1)*delup(k)+Sp(1,:)*dup'+s(N)*u(k-N+1);
   else
      ymod(k+1)=Sf(1,1)*delup(k)+Sp(1,:)*dup';
   end
   %disturbance compensation
   dist(k+1)=y(k+1)-ymod(k+1);
   dup=[delup(k),dup(1,1:N-3)];
end
%stairs plotting for input(zero order hold) and setpoint
[tt,uu]=stairs(tvec,u);
[ttr,rr]=stairs(tvec,r);
figure(1)
subplot(2,1,1)
plot(ttr,rr,'--',tvec,y(1:length(tvec)))
```

```
axis([0 kfinal*delt -2 2])
hold on
ylabel('y');
xlabel('time');
title('plant output(Output Temperature)');
subplot(2,1,2)
plot(tt,uu,'b')
axis([0 kfinal*delt -2 3])
hold on
ylabel('u');
xlabel('time');
title('Controller output(manipulated variable)');
figure(2)
plot(s,'ko')
xlabel('discrete time index, i');
ylabel('s(i)');
title('step response coefficients');
```

The output with the given Tuning parameters is shown in the following graphs:

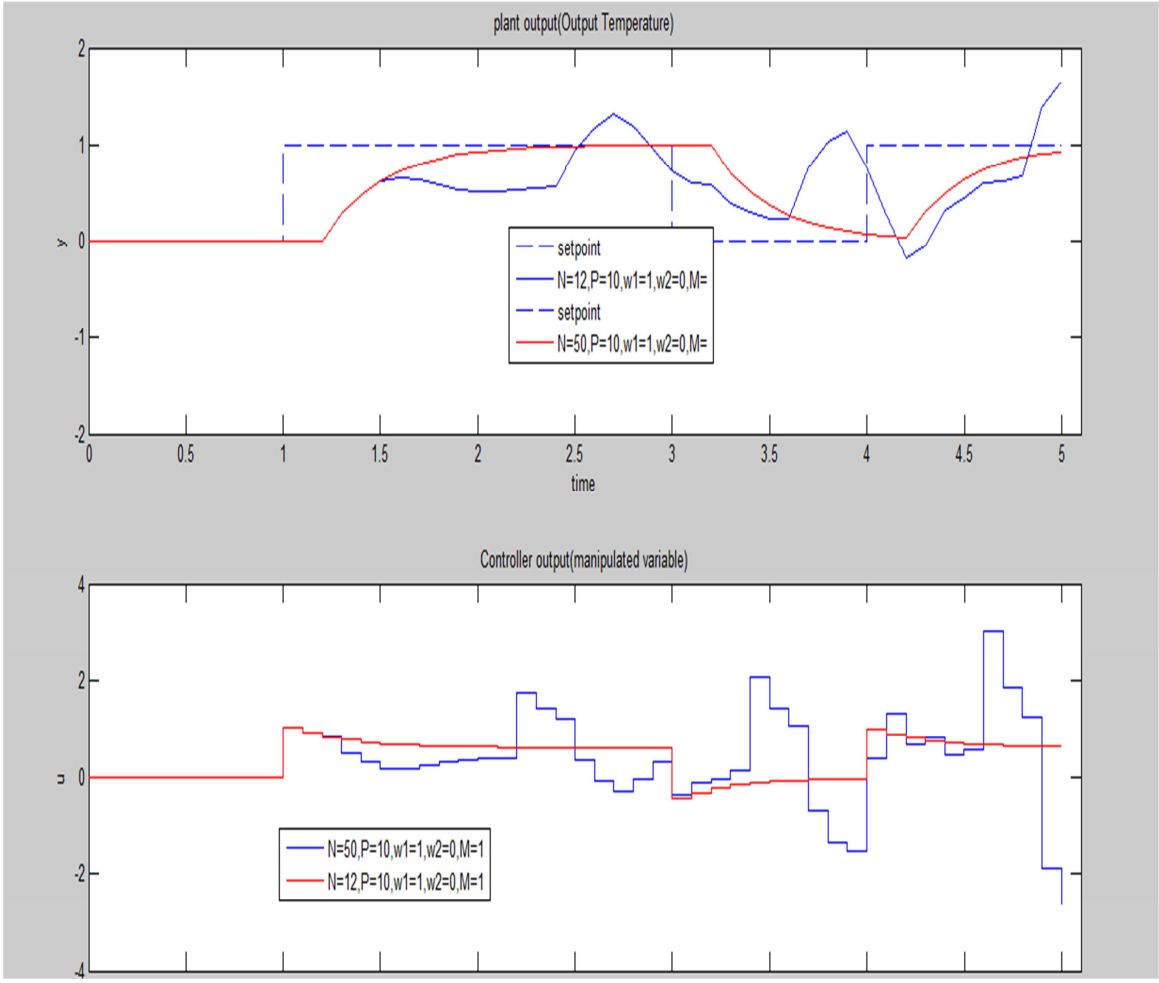


Simulation 4.3: The output of the DMC controlled plant (green). Control moves (blue solid). Set-point (blue dashed)

Now, we see the effects of various tuning parameters:

4.1.1 **Effect of Model Length N**

If we take two values of N=12 and N=50 keeping other parameters constant(P=10,M=1,w1=1,w2=0):

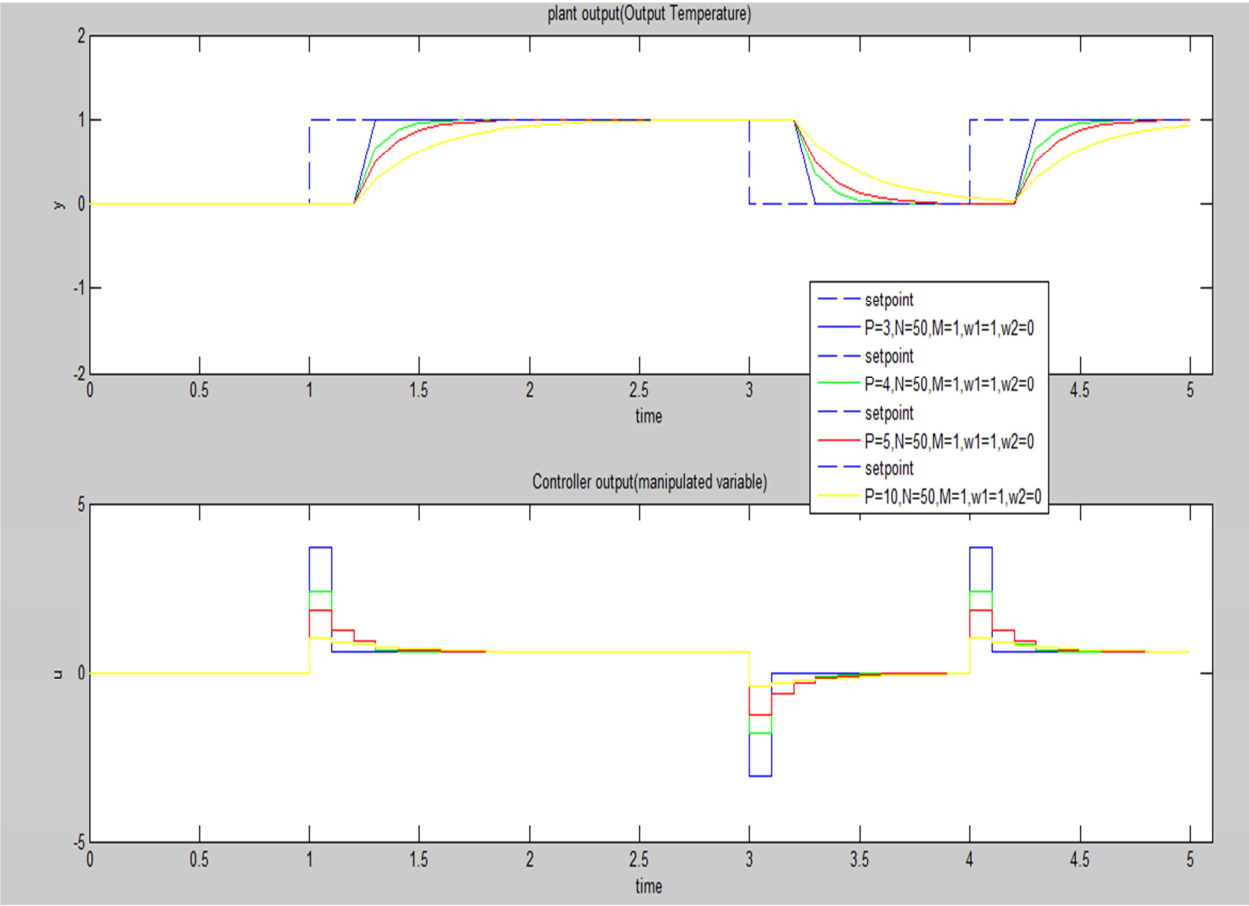


Simulation 4.4: The effect of model length N

Choosing a small Model length does not capture the dynamics of the process completely. This results in model error and poor performance.

4.1.2 **Effect of Prediction horizon P**

If we take different values of P keeping other parameters constant (P=50,M=1,w1=1,w2=0):

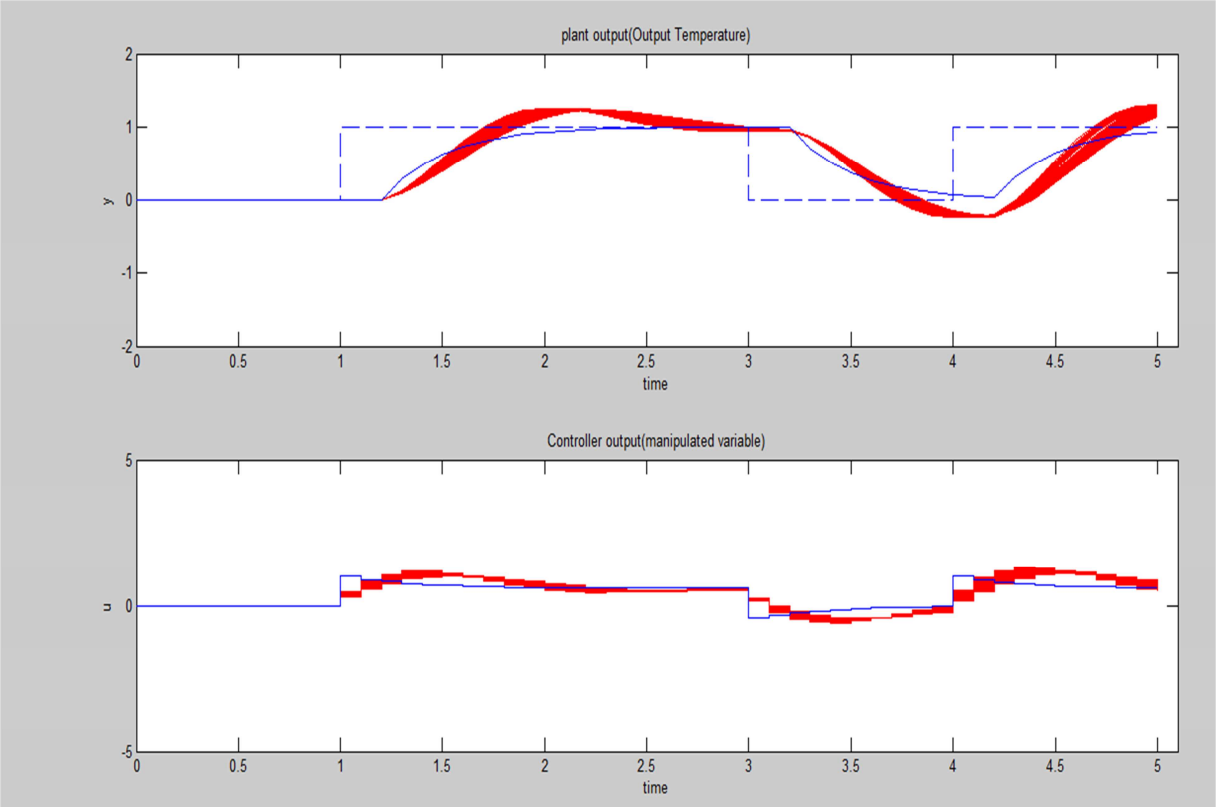


Simulation 4.5: The effect of Prediction Horizon P

A shorter Prediction Horizon will result in set-point being achieved very quickly. However, a shorter Prediction Horizon requires much more control action and is prone to modeling uncertainty.

4.3.3.1 Finding Suitable weight (w2) of control moves

Now we have taken w2=0 if P>>M. Now suppose we take two sets of Tuning parameters: P=10,M=1,w1=1,w2=0 and P=3,M=1,w1=1,w2=?. In second case M is not very much larger than M. We will now find the value of w2 such that the responses of both the cases are identical.



Simulation 4.6: Finding suitable value of weight w2

We vary w2 for P=3 from 0.70 and 0.90 (red lines) and compare it with the case when P=10 and w2=0(blue line) and find responses to be nearly identical.

4.1.3 Effect of noise:

There are numerous elements within the system which contribute to the measurement errors ranging from the problems of infrastructure to errors introduced in Transmission and collection of data from remote collation site. These errors are collated as additional inputs and thereby modify the response of the system in a random manner and are called as noise. These consist of random fluctuations about a mean value. These are taken as additional inputs after plant output in block diagram representation:

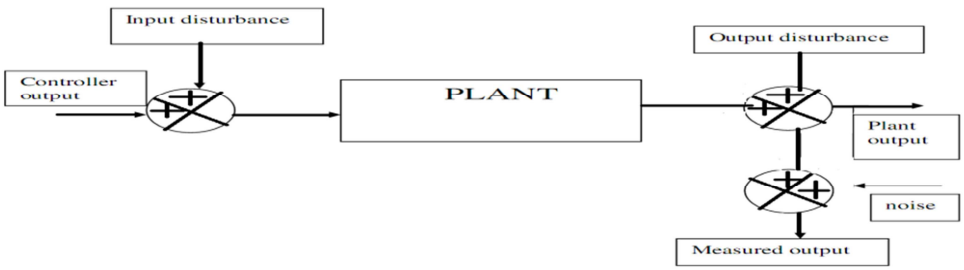
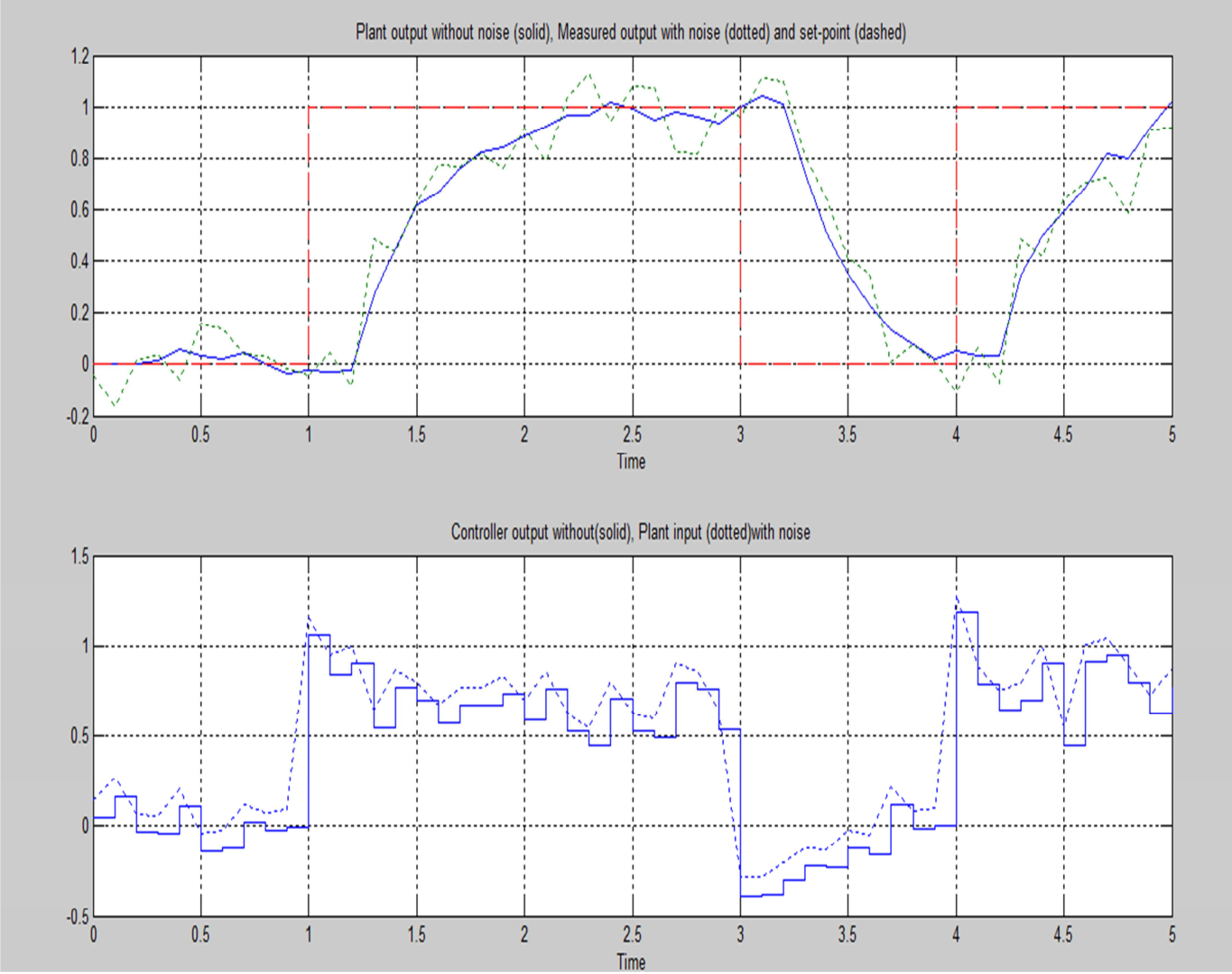


Fig 4.2: System with Noise

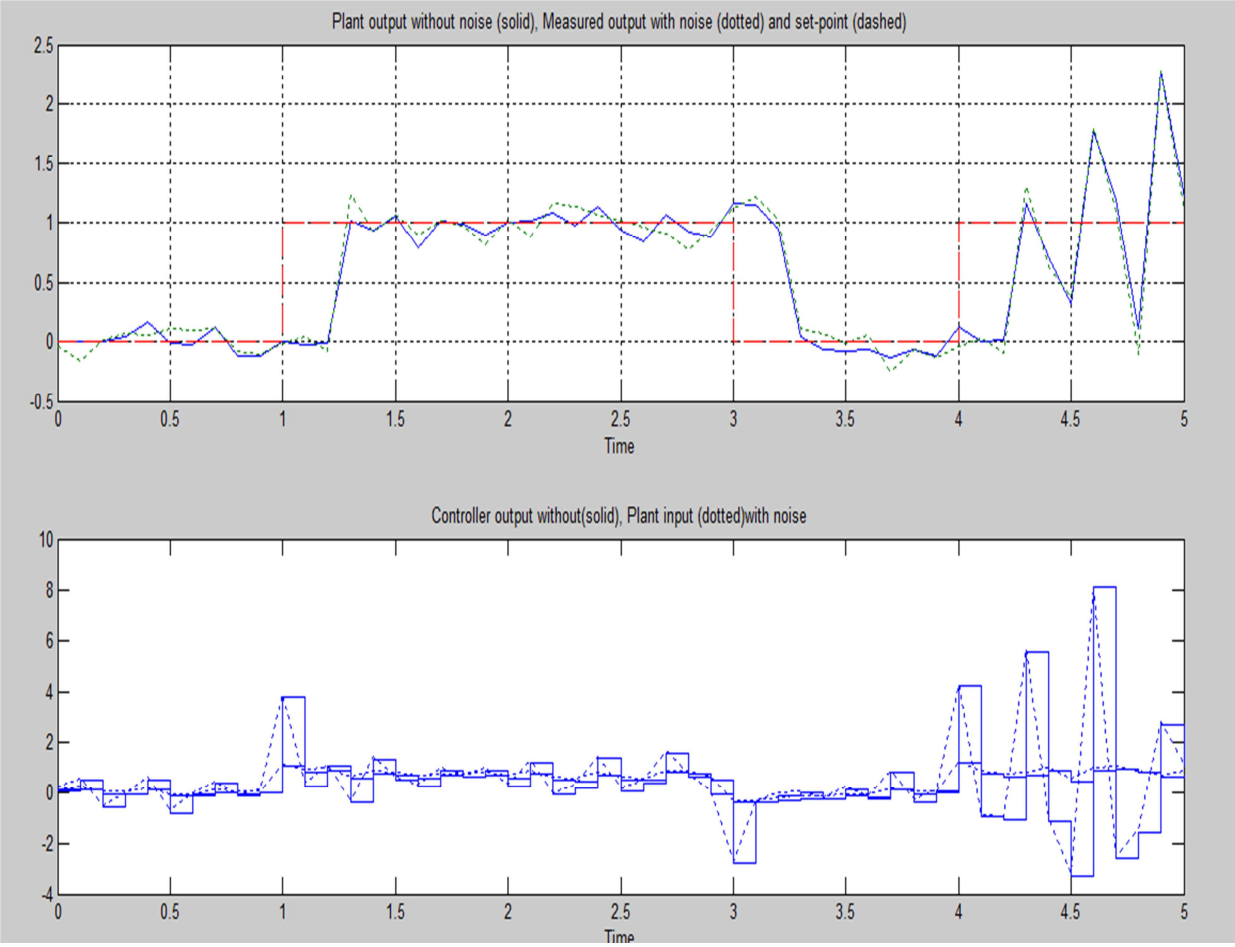
The output graph obtained through simulation is:



Simulation 4.7: Effect of noise on System Response

4.3.4.1 Effect of Control Horizon Length M on the effect of noise:

If we increase M from 1 to 5 and try to analyze the effect of noise on system as compared to the previous case, we come across following simulation**:**



Simulation 4.8: Effect of Control Horizon Length M on the effect of noise

From the above simulation result, it becomes clear that making M large makes the controller more sensitive to the effects of noise.

4.1.4 Effect of weighting Tuning parameters w1 and w2:

W1 is the weight of error vector while W2 is the weight of Control move vector. Let us study the effect of each of them.

Let us now consider, a process with the following process and disturbance transfer function:

$g(s) = 1/(5s^3+15.5s^2+11.5s+1)$ and $g_d(s)=1.52/50s+1$

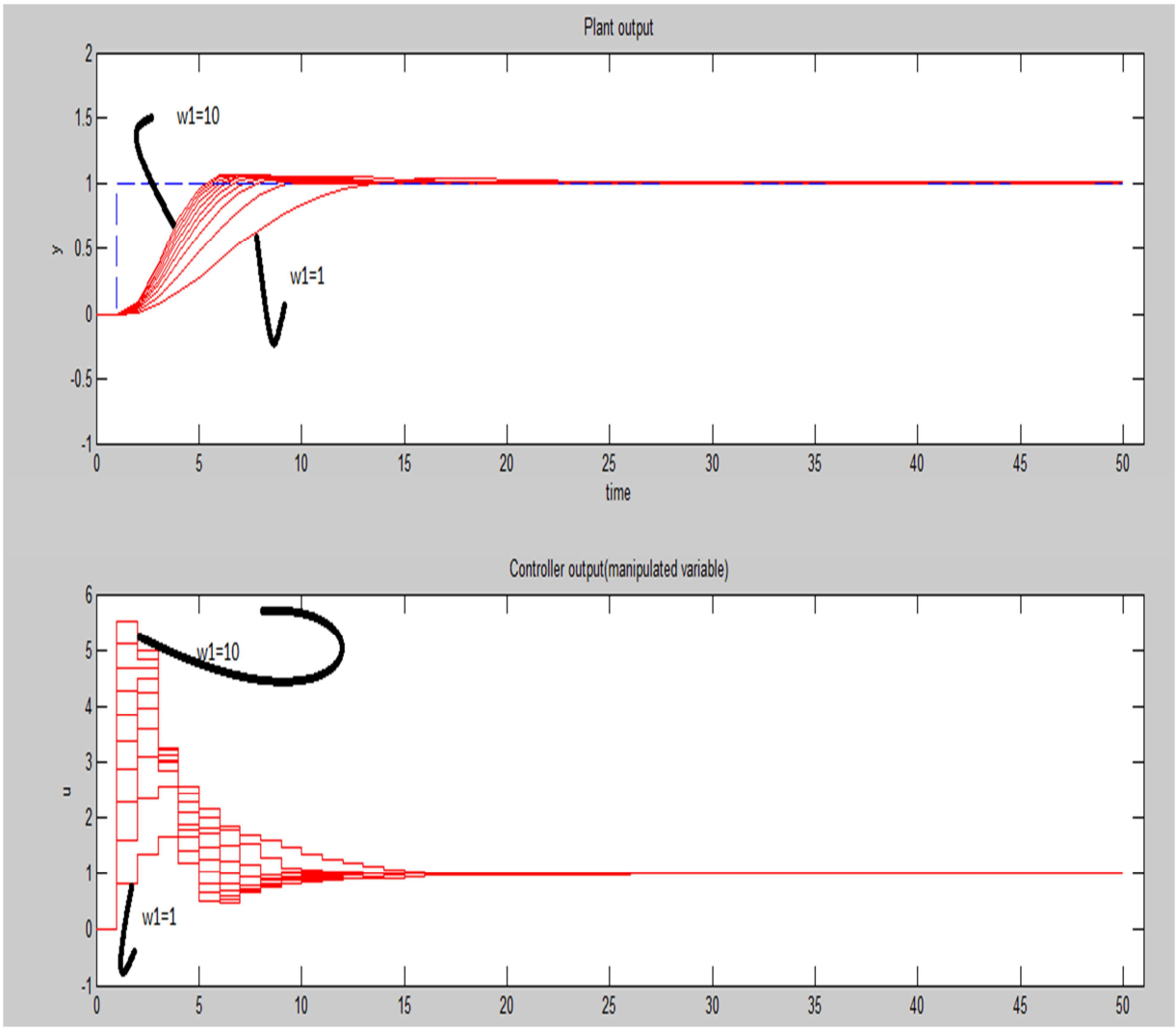
We shall implement a unconstrained DMC with horizon parameters set as P=15,M=5,and the following set of weights a) w1=1 and w2 varied from 1 to 10 b) w2=1 and w1 varied from 1 to 10

By using the stepinfo command in Matlab, we get the following information about step-response of process as well as the disturbance transfer function:

```
>> s=tf('s')
Transfer function:
s
>> sys_tf=1/(5*s^3+15.5*s^2+11.5*s+1)
Transfer function:
                1
  -------------------------
5 s^3 + 15.5 s^2 + 11.5 s + 1
>> S=stepinfo(sys_tf,'RiseTimeLimits',[0.1 0.9])
    S =
    Rise Time: 22.1961
    Settling Time: 40.6891
    Settling Min: 0.9030
    Settling Max: 0.9996
    Overshoot: 0
    Undershoot: 0
    Peak: 0.9996
    Peak Time: 78.9449
>>s=tf('s')
Transfer function:
s
>> sys_tf=1.52/(50*s+1)
Transfer function:
             1.52
            --------
            50 s + 1
>> S=stepinfo(sys_tf,'RiseTimeLimits',[0.1 0.9])
    S =
    Rise Time: 109.9146
    Settling Time: 195.6221
    Settling Min: 1.3723
    Settling Max: 1.5200
    Overshoot: 0
    Undershoot: 0
    Peak: 1.5200
    Peak Time: 524.5334
```

4.1.4.1 Effect of w1 :

Taking sampling interval as 1, we can take N=200 to cover the saturation of both process step response coefficients as well as the disturbance step response coefficients: w1 is varied from 1 to 10 keeping w1 at 1.



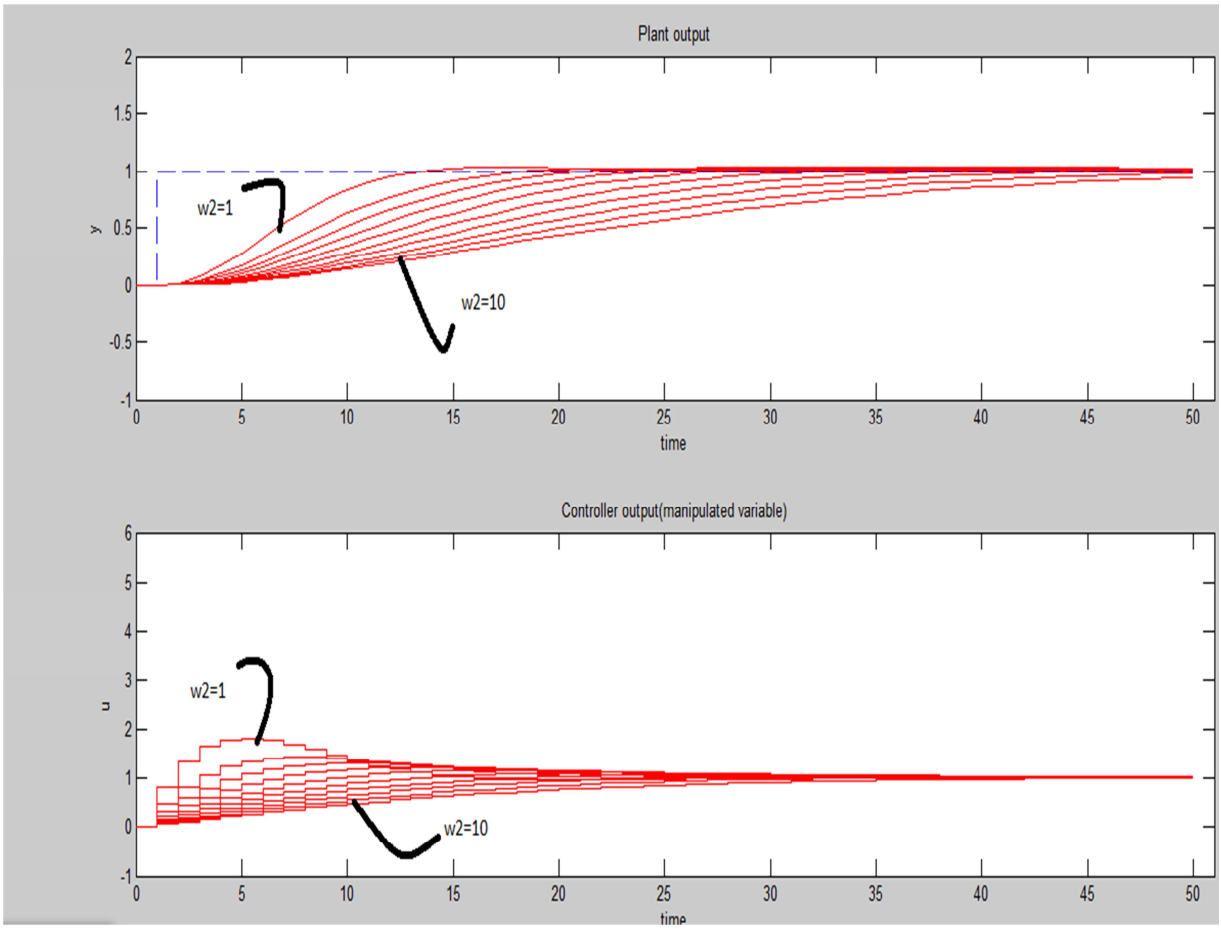
Simulation 4.9: Effect of error weight w1

We may see that increasing weight on error makes the response faster at the expense of more aggressive control

.

4.1.4.2 Effect of w2 :

Now w2 is varied from 1 to 10 keeping the weight w1 fixed at 1.Simulation in Matlab yields following graph:



Simulation 4.10: Effect of control move weight w2

We may see that increasing weight on control moves makes the response sluggish as penalty on control moves is being increased.

Chapter 5

MODELING OF A SISO SYSTEM (DC MOTOR)

5.1 The DC Motor:

We take a DC motor that dives an inertial with angular velocity GƟ(t) as the output and input voltage, $v_{in}(t)$, as the input. The purpose of this control system is to is to control the angular velocity of the motor by varying the input voltage. The following figure shows a simple model of the DC motor.

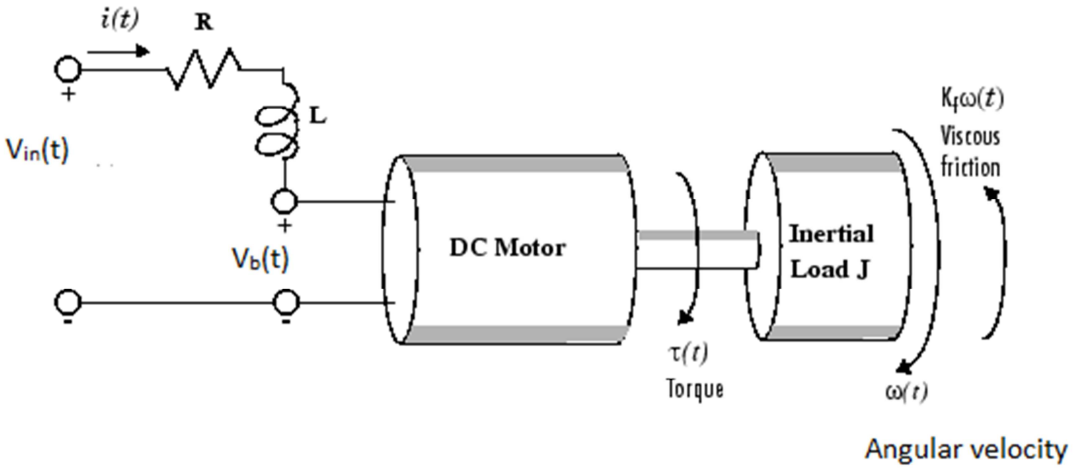


Figure 5.1: A Simple model for a DC Motor [16]

The strength of the magnetic field is assumed to be constant. The resistance of the circuit is denoted by R and the self-inductance of the armature by L.

5.2 Mathematical Equations*:*

The torque τ developed the shaft of the motor is proportional to the current *i* induced by the applied voltage,

$$\tau(t) = K_m i(t)$$

The back e.m.f, $V_b(t)$, is a voltage proportional to the angular velocity of the shaft

$$V_b(t)=K_bGʊ(t)$$

where K_m, the armature constant and Kb, the back emf constant depend on certain physical properties of the motor.

Mechanical equation is given as :

$$J\partial w/\partial t=\sum\tau_i=-K_f\omega(t)+K_mi(t) \quad \text{----(16)}$$

Where KfGʊ(t) denotes viscous frictional forces.

Electrical equation is given as :

$$V_{in}(t)=L\partial i/\partial t+Ri(t)+K_b\omega(t) \quad \text{----(17)}$$

Equations (17) & (16) can be rearranged as

$$\partial i/\partial t=-(R/L)i(t)-(K_b/L)\omega(t)+(1/L)V_{in}(t) \quad \text{----(18)}$$

$$\partial\omega/\partial t=(K_m/J)i(t)-(K_f/J)\omega(t) \quad \text{----(19)}$$

5.3 State-Space Equations for the DC Motor:

From equations (18) and (19) we can have the state space representation of the system in which current i and the angular velocity ω are the two state variables of the system. The input voltage $V_{in}(t)$, is the input to the system, and the angular velocity ω(t) is the output.

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega \end{bmatrix}=\begin{bmatrix} -\frac{R}{L} & -\frac{K_b}{L} \\ \frac{K_m}{J} & -\frac{K_f}{J} \end{bmatrix}\cdot\begin{bmatrix} i \\ \omega \end{bmatrix}+\begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix}\cdot V_{in}(t) \quad \text{----(20)}$$

The output equation is given by

$$y(t)=[0\ 1]\begin{bmatrix} i \\ w \end{bmatrix}+[0]V_{in}(t) \quad \text{----(21)}$$

Taking the following values of the parameters of the motor, we have

R=1.0 Ohms
L= 0.5 Henrys
Km = .015
Kb = .015
Kf = 0.2Nms
J= 0.02 kg.m^2

A = [-R/L -Kb/L; Km/J -Kf/J]=[-2 -0.03; 0.75 -10]
B = [1/L; 0]=[2;0]
C = [0 1];
D = [0];

Converting State Space Representation into Transfer Function using Matlab:

sys_dc = ss(A,B,C,D)
sys_tf = tf(sys_dc)

Transfer function is found out to be:

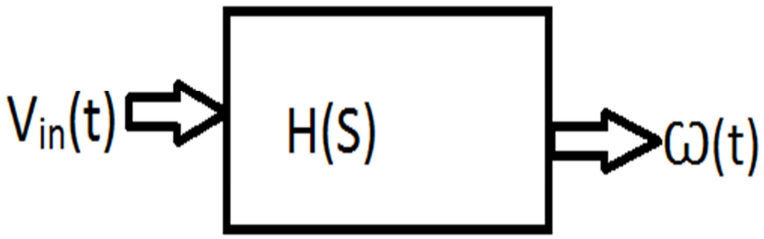
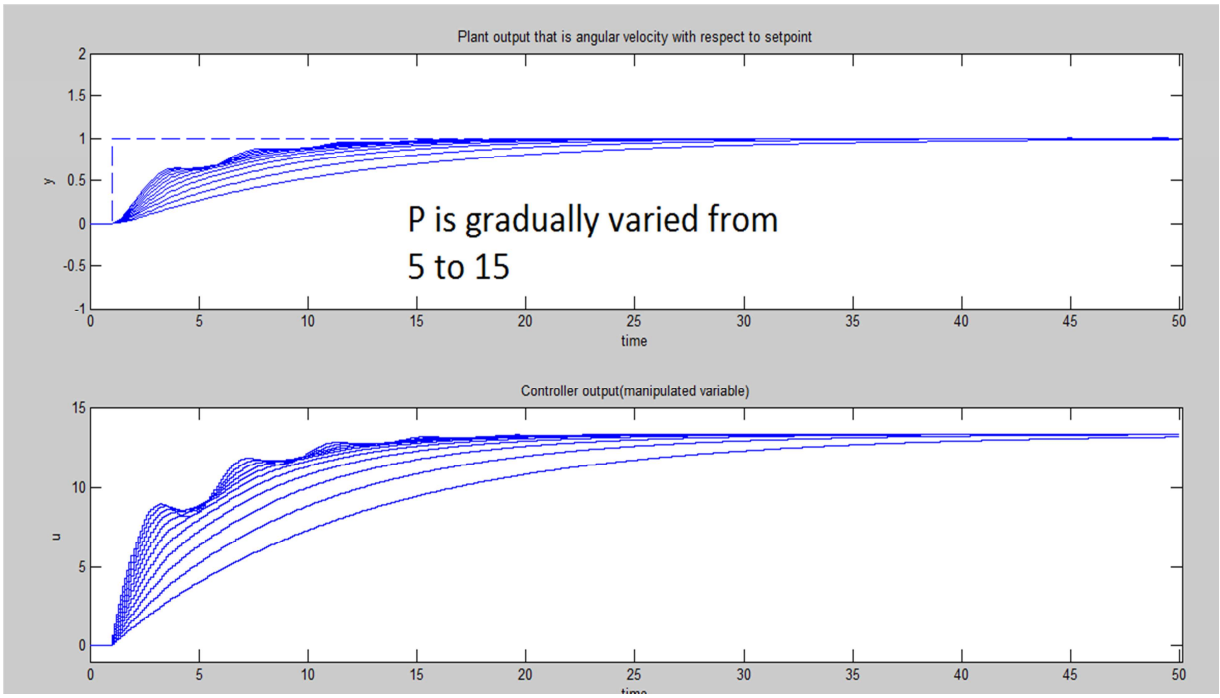


Figure 5.2: Block Diagram for transfer function of DC Motor

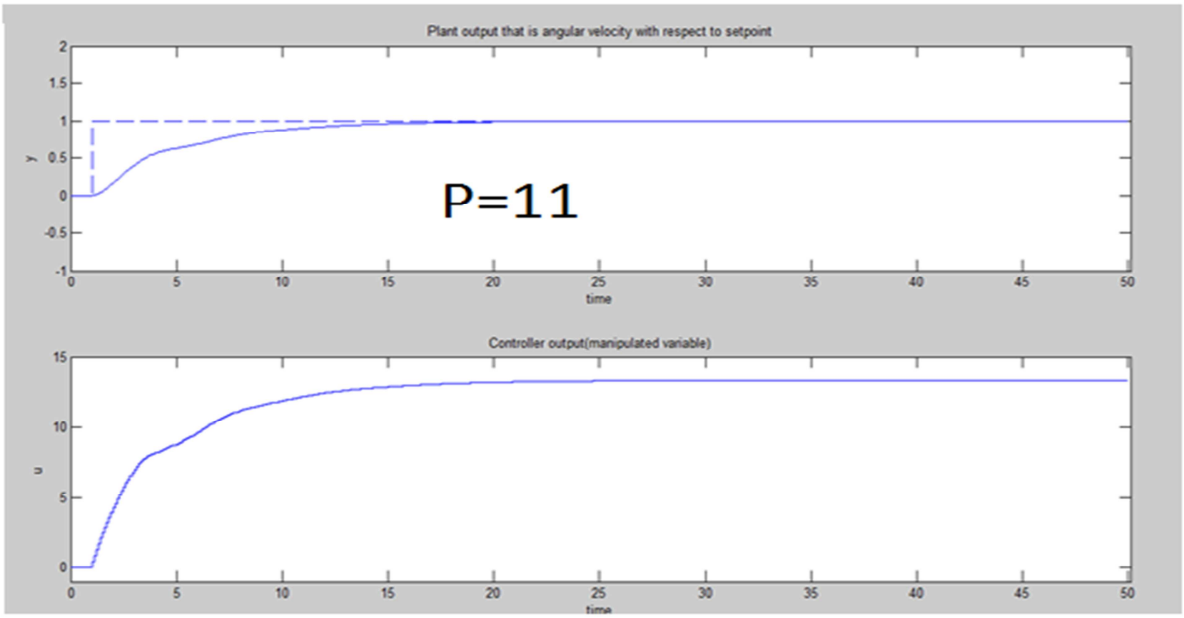
$$H(s) = \frac{1.5}{s^2 + 12\,s + 20.02}$$

The settling time of the above transfer function is 2.06650.Therefore, a model length N=30 and sampling interval Δt=0.1 will be taken up for our simulation. Taking w1=w2=1 and M=1, for various values of P(5 to 15) ,simulation is carried out .



Simulation 5.1: Effect of P on the DC Motor Response

From above simulation, we deduce that the best response is coming at P =11. This response is plotted separately in the following diagram:



Simulation 5.2: Best Response of DC Motor

Therefore, the optimum set of Tuning parameters of the modeled DC Motor is N=30, w1=w2=1 and M=1 and P=11.

Chapter 6

CONCLUSION AND FUTURE WORK

CHAPTER 6

6.1 Conclusion

MPC formulation has made it feasible to develop effective control strategies for constrained processes. During the course of this project, we have used a linear DMC for studying Model Predictive Control. The Design of any controller involves certain design parameters which are tuned to certain values to get the desired response. These parameters are known as the tuning Parameters. The tuning Parameters of DMC are mainly Model Length, Prediction Horizon Length, Control horizon Length and weighting factors. Simulations were performed in MatLab Environment for studying the effect of these parameters. This study helps in calculating optimum values of these parameters and this leads to the controlled variable suitably tracking the set-point.

Choosing a small Model length does not capture the dynamics of the process completely. This results in model error and poor performance. A shorter Prediction Horizon will result in set-point being achieved very quickly. However, a shorter Prediction Horizon requires much more control action and is prone to modeling uncertainty. We need not take a very long prediction horizon if a reasonably short prediction horizon can capture the dynamics of the process properly. For monotonous dynamic characteristics Control Horizon Length =1−2 while for oscillatory dynamic characteristics it is taken 4-8. The effect of noise on the system has also been described and it is verified that taking a small control Horizon makes the controller less sensitive to the effect of noise while increasing it increasing flexibility (but possibly at the cost of stability). Increasing weight on error makes the response faster at the expense of more aggressive control whereas increasing weight on control moves makes the response sluggish as penalty on control moves is being increased. These weighting parameters can drastically control the overall response and stability of the system. It has also been verified that DMC (MPC) has inbuilt compensation for systems with dead time and for systems exhibiting non-minimum phase behavior. Finally, we model a Single Input Single Output System in the form of a DC Motor and find out a suitable set of Tuning parameters for it.

6.2 Future work:

Some interesting works are yet to be done for furthering this research work. It will be interesting to implement this algorithm for a Multivariable System and study the interactions and couplings in the

system. There are many issues that can further be investigated. The impact of model uncertainty (robustness) on the performance on MPC is still largely unclear. Though there are certain practical approaches like varying the tuning parameters to minimize the effect of model uncertainty, this is far from a methodical solution to the challenge at hand. Further research can be taken up in the area of model identification, model determination and development of state estimation algorithms.

REFERENCES

[1], [3] P.E. Orukpe (2005). Basics of Model Predictive Control .ICM,EEE-CAP, London

[2], [9], [10] B. Wayne Bequette (2008). Process control: modeling, design, and simulation. Prentice Hall of India

[4], [5] Bela Liptak (2005).Process Control and Optimization. Taylor and Francis. London.

[6], [7] http://en.wikipedia.org/wiki/State_Space_Model

[8] C.R Cutler & B.C Ramaker (1979).Dynamic Matrix Control to Computer Control Algorithm Proceeding of the 86th National Meeting of the American Institute of Chemical Engineering, Houston, TX WP5-B

[11], [12] Zhijun Jiang, Xiaoling Huang (2009).The Character Study of Dynamic Matrix Control. International Conference on Intelligent Human-Machine Systems and Cybernetics

[13], [14], [15] Eduardo F. Camacho and Carlos Bordons. Model Predictive Control(1999). Springer-Verlag London Limited ,Graet Britain

[16] http://www.mathworks.com/help/toolbox/control/getstart/f1-1010549.html